

JOURNAL OF COMPLEXITY **13**, 180–193 (1997)
ARTICLE NO. CM970439

Decoding of Reed Solomon Codes beyond the Error-Correction Bound

Madhu Sudan*

IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598

Received August 31, 1996

We present a randomized algorithm which takes as input n distinct points $\{(x_i, y_i)\}_{i=1}^n$ from $F \times F$ (where F is a field) and integer parameters t and d and returns a list of all univariate polynomials f over F in the variable x of degree at most d which agree with the given set of points in at least t places (i.e., $y_i = f(x_i)$ for at least t values of i), provided $t = \Omega(\sqrt{nd})$. The running time is bounded by a polynomial in n . This immediately provides a maximum likelihood decoding algorithm for Reed Solomon Codes, which works in a setting with a larger number of errors than any previously known algorithm. To the best of our knowledge, this is the first efficient (i.e., polynomial time bounded) algorithm which provides error recovery capability beyond the error-correction bound of a code for any efficient (i.e., constant or even polynomial rate) code. ©1997 Academic Press

1. INTRODUCTION

1.1. Problem Statement

We consider the following problem.

PROBLEM 1.

Input: A field F ; n distinct pairs of elements $\{(x_i, y_i)\}_{i=1}^n$ from $F \times F$; and integers d and t .

Output: A list of all functions $f : F \rightarrow F$ satisfying

$$\left. \begin{array}{l} f(x) \text{ is a polynomial in } x \text{ of degree at most } d \\ \text{with } |\{i | f(x_i) = y_i\}| \geq t. \end{array} \right\} \quad (1)$$

The above problem and close variants have been considered before in the context of coding theory and learning. The best threshold on t for which

*E-mail: madhu@watson.ibm.com.

a polynomial time bounded algorithm solved this problem was previously $t \geq (n + d + 1)/2$. Note that the ratio $t/n \rightarrow \frac{1}{2}$ if we fix d and let $n \rightarrow \infty$. In fact, when t satisfies this property then there is a unique function f satisfying (1). A particularly simple algorithm for this case is given by Berlekamp and Welch [4] (see, for instance, Gemmell and Sudan [9]).

In this paper we present an algorithm which solves the problem given above for $t = \Omega(\sqrt{nd})$. Note that for fixed d , as $n \rightarrow \infty$, the fraction of agreement required by our algorithm approaches 0 (and not $\frac{1}{2}$ which is what previous algorithms seemed to get). The algorithm is based on an algorithm of Ar *et al.* [1] for a restricted version of this problem.

The task of reconstructing polynomials describing a given set of points is relevant in the context of coding theory and we describe this context next. This task may also be of some relevance to computational complexity theory. We touch on this motivation very briefly in the conclusions.

1.2. Error-Correcting Codes

For integers n, k, δ and a collection of symbols Σ , an $[n, k, \delta]$ -code over Σ is a collection $\mathcal{C} \subset \Sigma^n$ of n -letter words over the alphabet Σ with $|\mathcal{C}| = |\Sigma|^k$ and the property that any two strings in \mathcal{C} differ in at least δ places (i.e., the strings have Hamming distance δ). Given a code \mathcal{C} , the largest δ for which \mathcal{C} is a $[n, k, \delta]$ -code is referred to as the *distance* of the code. If τ satisfies $2\tau + 1 \leq \delta$, then an $[n, k, \delta]$ -code \mathcal{C} is also a τ -error-correcting code. The largest such value τ will be referred to as the *error-correction bound* of a code. This terminology reflects the fact that, given any string $s \in \Sigma^n$, there is at most one string $c \in \mathcal{C}$ which is within a Hamming distance of τ from s .

The codes of relevance to this paper are the Reed Solomon Codes. For a finite field F of size n and parameter d , the *Reed Solomon Code* is an $[n, d + 1, n - d]$ -code over F , whose codewords are the strings $\{(p(0), p(w), p(w^2), \dots, p(w^{n-1}))\}_p$, where w is some fixed primitive element of F and p ranges over all polynomials of degree at most d over F (see, for instance, [18, page 86]).

The algorithmic tasks of relevance to this paper are the tasks of “error-correction” and “maximum-likelihood decoding.” The problem of τ -error-correction for an $[n, k, \delta]$ -code is defined for $2\tau + 1 \leq \delta$ as follows: “Given a string $s \in \Sigma^n$, find a string $c \in \mathcal{C}$ which is within Hamming distance τ of s , if one such exists.” Since \mathcal{C} is a τ error-correcting code, the answer, if it exists, is unique. Note that the problem is not defined for values of τ and δ (i.e., when $2\tau + 1 > \delta$) which may allow for non-unique answers. The *maximum-likelihood decoding problem* (see, for instance, [17, page 8]) is set in a model where a *larger number of errors is considered less likely* and is defined as follows: “Given a string $s \in \Sigma^n$, find a (the) codeword $c \in \mathcal{C}$ which is nearest to s (i.e., least Hamming distance away from s).” This problem is also sometimes referred to as the *nearest codeword problem*.

Previous work has focused mainly on the task of error-correction, and algorithms are known for error-correction of many interesting codes. In particular, for Reed Solomon Codes the τ -error-correction problem for a τ -error correcting code can be solved in polynomial time. In particular, the earlier mentioned solution of Berlekamp and Welch [4] works in this setting.

The case of recovering from an error larger than the error-correction capacity of a code has not attracted the same amount of attention and significantly less is known about this problem. Since in this case the solution to the maximum-likelihood decoding problem may not be unique, it is not clear which solution is to be reported when the answer is not unique. Further, it is not clear why any algorithm would be able to (or should be allowed to) prefer any one solution over any other equally respectable solution.

However, it is possible to define a closely related problem which does not offer the algorithm any choice in its solutions. This problem, sometimes called the τ -reconstruction problem, is defined as follows: “Given a string s , find *all* codewords $c \in \mathcal{C}$ that are within Hamming distance τ of s .” This problem is also known as the *list decoding problem*. The reconstruction problem offers the solution to the maximum-likelihood decoding problem for a much larger range of τ than is allowed by the τ -error correction problem. This is the problem tackled in this paper for the case of Reed Solomon Codes.

The τ -reconstruction problem is not a universal panacea for the maximum-likelihood problem. In fact, by making the task enumerative (rather than picking one element from a large set, we want the whole set), the complexity requirements of the task go up. In particular, the running time is lower bounded by the output size. Bounds on the output size of the reconstruction problem have been studied in the context of coding theory and a well known bound, due to Johnson (see [17, page 525]), bounds the number of solutions by $n(n-\delta)/(2\tau^2 - (2n)\tau + n(n-d))$ for binary codes (i.e., codes over the alphabet $\{0, 1\}$), provided that the denominator is positive. For general codes, a simple bound can be shown by an inclusion-exclusion argument (see, for instance, [11]) which yields that the number of solutions to the τ -reconstruction problem is at most $2/\varepsilon$, if $\tau = (1 - \varepsilon)\delta$, provided $\tau < n - \sqrt{2n(n - \delta)}$. (Another such bound is also known due to Goldreich, *et al.* [11]. We do not describe this here.) However the inclusion-exclusion bound is not constructive, i.e., it does not provide a list of the $2/\varepsilon$ codewords which may be the solution to the reconstruction problem.

It is reasonable to ask for a solution to the reconstruction problem which runs in polynomial time, when the output size is bounded. Here we solve the τ -reconstruction problem for Reed Solomon Codes for exactly the same values of the parameters δ and τ for which the inclusion-exclusion bound seems to work. Finding bounds which work for more general settings of δ and τ and finding solutions to the reconstruction problem which work in such settings remain open questions.

1.3. Previous Work

As mentioned in the previous section, the τ -error-correcting problem is well-studied and we will not describe past work in that problem here. The definition of the τ -reconstruction problem used here is based on definitions of Ar *et al.* [1] and Goldreich *et al.* [11]. To the best of our knowledge, there are only two instances where the τ -reconstruction problem has found interesting solutions for some error-correcting code. The first, due to Goldreich and Levin [10], provides a solution for certain families of Hadamard Codes. Kushilevitz and Mansour [15] provide a variant of this algorithm which applies to the same codes. The second instance involves a generalization of the codes and algorithm given by [10] and is due to [11]. Both the codes given here are extremely low-rate codes. In fact, the rate (i.e., the ratio k/n) for these codes is $O(\log^{O(1)} n/n)$ and thus a brute-force algorithm (running in time $|\Sigma|^{O(k)}$) is not too inefficient for these problems. The feature that makes the solutions of [10, 11, 15] interesting and efficient is that they work in time polynomial in k , using random access to an oracle describing the input s . This makes the solution interesting in some learning theoretic and cryptographic settings, but is however not very useful for coding theoretic applications (due to the low information rate). The techniques of Goldreich and Levin (which are inherited by [11, 15]) are interesting in that they manage to convert, modularly, the nonconstructive bounds on the number of outputs (discussed earlier) into constructive ones. But their technique does not appear to generalize to the setting of Reed Solomon Codes (or any other high-rate codes).

Ar *et al.* [1] do provide some solutions to the reconstruction problem, but not in its full generality. In particular, they restrict the nature of the input word s . For this restricted case they provide a solution to the reconstruction problem based on algebraic techniques (and in particular uses polynomial time solutions to the bivariate factoring problem). Our solution is a minor modification of their algorithm and analysis which manages to get around their restriction.

2. ALGORITHM

We now present our algorithm for solving the problem given in Section 1.1.

DEFINITION 1 (Weighted Degree). For weights $w_x, w_y \in \mathbb{Z}^+$, the (w_x, w_y) -weighted degree of a monomial $q_{ij}x^i y^j$ is $iw_x + jw_y$. The (w_x, w_y) -weighted degree of a polynomial $Q(x, y) = \sum_{ij} q_{ij}x^i y^j$ is the maximum, over the monomials with nonzero coefficients, of the (w_x, w_y) -weighted degree of the monomial.

ALGORITHM. /* Inputs: $n, d, t; \{(x_1, y_1), \dots, (x_n, y_n)\}$ */

A. /* Parameters l, m to be set later. */

B. Find *any* function $Q : F^2 \rightarrow F$ satisfying

$$\left. \begin{array}{l} Q(x, y) \text{ has } (1, d) - \text{weighted degree at most } m + ld, \\ \forall i \in [n], Q(x_i, y_i) = 0, \\ Q \text{ is not identically zero.} \end{array} \right\} \quad (2)$$

C. Factor the polynomial Q into irreducible factors.

D. Output all the polynomials f such that $(y - f(x))$ is a factor of Q and $f(x_i) = y_i$ for at least t values of i from $[n]$.

Note. Step C above can be solved in randomized polynomial time with zero-sided error. If F is of characteristic zero, or if the running time is allowed to be polynomial in $|F|$, then the solution can be obtained deterministically. (See, for instance, [13].)

3. ANALYSIS

In order to prove that the algorithm above can be run in polynomial time and works correctly, we need to prove the following set of claims. In all the following claims, we fix the set of pairs $\{(x_1, y_1), \dots, (x_n, y_n)\}$.

CLAIM 2. *If a function $Q : F^2 \rightarrow F$ satisfying (2) exists, then one can be found in time $\text{poly}(n)$.*

Proof. Let $Q(x, y) = \sum_{j=0}^l \sum_{k=0}^{m+(l-j)d} q_{kj} x^k y^j$. Then we wish to find coefficients q_{jk} satisfying the constraints $\sum_{j=0}^l \sum_{k=0}^{m+(l-j)d} q_{kj} x_i^k y_i^j = 0$, for every $i \in [n]$. This is a linear system in the unknowns $\{q_{kj}\}$ and hence if a solution exists then one can be found in polynomial time. ■

CLAIM 3. *If $(m+1)(l+1) + d \binom{l+1}{2} > n$ then there exists a function $Q(x, y)$ satisfying (2).*

Proof. First we observe that the linear system is homogeneous. Therefore the setting $q_{kj} = 0$ satisfies all the linear constraints. However, this does not satisfy (2), since Q would be identically zero. In order to show that a nonzero solution exists, we observe that the number of unknowns in the linear system that we wish to solve is $(m+1)(l+1) + d \binom{l+1}{2}$. Since this is more than n , we have a homogeneous linear system with more variables than constraints and hence a nonzero solution exists. ■

The following lemma is a special case of a general class of theorems known as Bezout's Theorem. Since we will be interested in tight behavior of the theorem, we use a version due to [1]. The proof is also from [1].

CLAIM 4. *If $Q(x, y)$ is a function satisfying (2) and $f(x)$ is a function satisfying (1) and $t > m + ld$, then $(y - f(x))$ divides $Q(x, y)$.*

Proof. Consider the function $p(x) \stackrel{\text{def}}{=} Q(x, f(x))$. This is a polynomial in x and we argue that it has degree at most $m + ld$. Consider any monomial $q_{kj}x^ky^j$ of $Q(x, y)$. Since Q has $(1, d)$ -weighted degree at most $m + ld$, we have that $k + jd \leq m + ld$. Thus the term $q_{kj}x^k(f(x))^j$ is a polynomial in x of degree at most $k + jd \leq m + ld$. Thus $p(x)$ has degree at most $m + ld$ (since it is a sum of polynomials of degree at most $m + ld$).

Now we argue that $p(x)$ is identically zero. Since $Q(x_i, f(x_i))$ is zero whenever $y_i = f(x_i)$, we have that $p(x_i)$ is zero for strictly greater than $m + ld$ points. Thus p has more zeroes than its degree and hence is identically zero, implying $Q(x, f(x)) \equiv 0$.

Now consider $Q(x, y)$ as a polynomial $Q_x(y)$ in y with coefficients from $F[x]$, the ring of polynomials in x . By the polynomial remainder theorem, we have that if $Q_x(\zeta) = 0$ then $(y - \zeta)$ divides $Q_x(y)$. Substituting $\zeta = f(x)$ yields the lemma. ■

We are now ready to optimize the parameters m and l . For now we will ignore the fact that l and m have to be integers and fix this aspect later. Note that we want

$$m + ld < t \quad \text{and} \quad (m + 1)(l + 1) + d \binom{l+1}{2} > n.$$

Thus, given a value of l , we can compute the smallest m for which the second condition holds, and that is $(n + 1 - d \binom{l+1}{2}) / (l + 1) - 1$. Thus we find that t must be at least

$$\frac{n + 1 - d \binom{l+1}{2}}{l + 1} - 1 + ld + 1 = \frac{n + 1}{l + 1} + \frac{dl}{2}.$$

We can now minimize the expression above as a function of the unknown parameter l to obtain the smallest value of t for which this algorithm will work. The minimum occurs when

$$\frac{-(n + 1)}{(l + 1)^2} + \frac{d}{2} = 0 \Rightarrow l = \sqrt{\frac{2(n + 1)}{d}} - 1.$$

This setting yields

$$m \geq \sqrt{(n + 1)d/2} - \sqrt{(n + 1)d/2} + d/2 - 1 = d/2 - 1$$

and

$$t > m + ld \geq d/2 - 1 + \sqrt{2(n + 1)d} - d = \sqrt{2(n + 1)d} - d/2 - 1.$$

Due to the integrality issues we will lose a little bit in the following theorem.

THEOREM 5. *Given a sequence of n distinct pairs $\{(x_i, y_i)\}_{i=1}^n$, where the x_i s and y_i s are elements of a field F , and integer parameters t and d such that $t \geq d\lceil\sqrt{2(n+1)/d}\rceil - \lfloor d/2 \rfloor$, there exists an algorithm, which runs in time polynomial in n , that can find all the polynomials $f : F \rightarrow F$ of degree at most d , such that the number of points (x_i, y_i) such that $y_i = f(x_i)$ is at least t .*

Proof. We use the algorithm of Section 2 with $m = \lceil d/2 \rceil - 1$ and $l = \lceil \sqrt{(2(n+1)/d)} \rceil - 1$. It may be verified that this setting satisfies the condition $(m+1)(l+1) + d\binom{l+1}{2} \geq n+1$. Hence, by Claim 3, Step 2 will return a function $Q(x, y)$ satisfying property (2). Furthermore, the setting of m and l also satisfies the condition $t > m + ld$. Thus Claim 4 guarantees that if f is a function satisfying (1), then $(y - f(x))$ will divide the polynomial Q returned in Step 2 and hence be one of our outputs. ■

4. BOUND ON THE NUMBER OF POLYNOMIALS

Here we give a new proof of an upper bound given in [11] on the number of polynomials f of degree d agreeing with t out of n distinct points $\{(x_i, y_i)\}$. Their proof uses an inclusion–exclusion argument, while ours is different. Note that their bound is exactly the same and applies under exactly the same conditions on n , t , and d , with the important difference being that our proof holds only for the univariate polynomial case, while theirs applies more generally. Nevertheless, we feel that this new proof may be of some interest. Furthermore, this justifies the statement, made in Section 1, that our algorithm works in exactly the same setting as the inclusion–exclusion bound.

LEMMA 6. *If $t/n \geq (\sqrt{2 + d/4n} \cdot \sqrt{d/n}) - d/2n$, then the number of polynomials f of degree d satisfying (1) is at most*

$$\left\lfloor \frac{t}{d} + \frac{1}{2} - \sqrt{\left(\frac{t}{d} + \frac{1}{2}\right)^2 - \frac{2n}{d}} \right\rfloor \leq \frac{2n}{t + d/2}.$$

Remark. The bound above shows that when t is linear in n , then the number of polynomials is bounded by a constant (see also Table I). In particular, if $t \geq (n+d)/2$, (i.e., the number of errors is less than the error-correction limit), then the bound shown above is 1.

Proof. If m and l are integers such that the algorithm of the previous section works correctly, then the algorithm of Section 2 gives at most l solutions

TABLE I

Rate $\stackrel{\text{def}}{=} \frac{k}{n}$	Error-correction bound	Error tolerated by our algorithm	# Solns reported by our algorithm
.5	.25	.25	1
.33	.333	.35	2
.25	.375	.417	2
.10	.45	.602	4

implying that l is an upper bound on the number of polynomials of degree d which can agree with n given points at t places. In other words, if m and l are integers satisfying

$$(m+1)(l+1) + d \binom{l+1}{2} > n \quad (3)$$

and

$$m + ld + 1 \leq t, \quad (4)$$

Then l is an upper bound on the number of functions satisfying (1). Equation (4) indicates that we should pick m to be as large as possible subject to the constraint $m \leq t - ld - 1$. We therefore set m to $t - ld - 1$. Thus (3) reduces to

$$(t - ld)(l + 1) + d(l + 1)l/2 > n.$$

In other words, we require

$$\left(\frac{d}{2}\right)l^2 - \left(t - \frac{d}{2}\right)l + (n - t) < 0. \quad (5)$$

Let

$$\alpha \stackrel{\text{def}}{=} \frac{1}{d} \left((t - d/2) - \sqrt{(t - d/2)^2 - 4(d/2)(n - t)} \right)$$

and

$$\delta \stackrel{\text{def}}{=} \frac{2}{d} \left(\sqrt{(t - d/2)^2 - 4(d/2)(n - t)} \right).$$

Then α and $\alpha + \delta$ are the two roots to (5). If $\delta > 1$, then $l = \lfloor \alpha + 1 \rfloor^1$ and

¹Note that we need l to be an integer which satisfies the inequality (5) strictly and hence we are forced to use $\lfloor \alpha + 1 \rfloor$ rather than just $\lceil \alpha \rceil$.

$m = t - ld - 1$ satisfy conditions above and l provides an upper bound on the number of functions satisfying (1).

The condition $\delta > 1$ is satisfied if

$$\begin{aligned}
 & \sqrt{\left(\frac{2t}{d} - 1\right)^2 - 8\frac{n}{d} + 8\frac{t}{d}} > 1. \\
 & \Leftrightarrow \sqrt{\left(\frac{2t}{d} + 1\right)^2 - 8\frac{n}{d}} > 1. \\
 & \Leftrightarrow \left(\frac{2t}{d} - 1\right)^2 - 8\frac{n}{d} > 1. \\
 & \Leftrightarrow \frac{2t}{d} + 1 > \sqrt{8\frac{n}{d} + 1}. \\
 & \Leftrightarrow t > \frac{d}{2} \left(-1 + \sqrt{8\frac{n}{d} + 1}\right) = n \left(\sqrt{\frac{d}{n}} \sqrt{2 + \frac{d}{4n}} - \frac{d}{2n}\right).
 \end{aligned}$$

If t satisfies the condition above then we get the following bound for l :

$$\begin{aligned}
 l &= \lfloor \alpha + 1 \rfloor \\
 &= \left\lfloor \left(\frac{t}{d} - \frac{1}{2}\right) - \sqrt{\left(\frac{t}{d} - \frac{1}{2}\right)^2 - \frac{2(n-t)}{d}} + 1 \right\rfloor \\
 &= \left\lfloor \left(\frac{t}{d} + \frac{1}{2}\right) - \sqrt{\left(\frac{t}{d} + \frac{1}{2}\right)^2 - \frac{2n}{d}} \right\rfloor.
 \end{aligned}$$

This yields the lemma. (Equation [11] already shows that the final quantity is upper bounded by $2n/(t + d/2)$, so we do not have to show that part.) ■

5. EXTENSIONS TO MULTIVARIATE POLYNOMIALS

It is relatively simple to extend the algorithm and the analysis of the earlier sections directly to apply for multivariate polynomial fitting.

We first extend the problem definition. Some slight care has to be taken to determine what is an appropriate extension of the problem definition, and the definition below turns out to be the one for which the extension works easily.

We consider k -variate polynomials. We use $x^{(1)}, \dots, x^{(k)}$ to denote the k variables. The shorthand \hat{x} will be used to denote this tuple of variables in vector notation.

PROBLEM 2.

Input: A field F , a set $H \subset F$, a function $f': H^k \rightarrow F$, and integers t and d .

Output: A list of all functions $f: F^k \rightarrow F$ satisfying

$$\left. \begin{array}{l} f(\hat{x}) \text{ is a polynomial in } \hat{x} \text{ of degree at most } d \\ \text{and } |\{\hat{x} \in H^k | f(\hat{x}) = f'(\hat{x})\}| \geq t. \end{array} \right\} \quad (6)$$

The algorithm is a straightforward generalization of the algorithm of Section 2. We first extend the definition of weighted degree in the obvious way.

DEFINITION 7. For integers w_1, \dots, w_n , the (w_1, \dots, w_n) -weighted degree of a monomial $\prod_{i=1}^n x_i^{d_i}$ is $\sum_i w_i d_i$. The (w_1, \dots, w_n) -weighted degree of a polynomial $Q(x_1, \dots, x_n)$ is the maximum, over the monomials with nonzero coefficients in Q , of the (w_1, \dots, w_n) weighted degree of the monomial.

MULTIVARIATE ALGORITHM. /* Inputs: $d, t, k; H \subset F; \{f'(x)\}_{x \in H^k}$ */

A'. /* Parameters l, m to be set later. */

B'. Find any function $Q: F^{k+1} \rightarrow F$ satisfying

$$\left. \begin{array}{l} Q(\hat{x}, y) \text{ has } (1, \dots, 1, d)\text{-weighted degree at most } m + ld, \\ \forall \hat{x} \in H^k, Q(\hat{x}, f'(\hat{x})) = 0, \\ Q \text{ is not identically zero.} \end{array} \right\} \quad (7)$$

C'. Factor the polynomial Q into irreducible factors.

D'. Output all degree d polynomials f such that $(y - f(\hat{x}))$ is a factor of Q and $f(\hat{x}) = f'(\hat{x})$ for at least t values of $\hat{x} \in H^k$.

As usual we need to ensure that the number of coefficients of Q is more than n and prove that, for sufficiently large t , the algorithm will output all solutions to the reconstruction problem above.

CLAIM 8. If $m + ld \geq (k + 1)(d + 1)^{1/(k+1)} h^{k/(k+1)}$, then there exists a function $Q(\hat{x}, y)$ satisfying (7).

Proof (Sketch). The number of coefficients in Q is strictly larger than $(1/(m + d + 1)) \binom{m + ld + k + 1}{k + 1}$. (The bound stated is a gross approximation. A generic degree $m + ld$ polynomial in $k + 1$ variables has $\binom{m + ld + k + 1}{k + 1}$ coefficients. Q is a degree $m + ld$ polynomial in $k + 1$ variables with the restriction that the degree of y only ranges from 0 to l . This restriction is taken care of by the

$1/(m+d+1)$ factor.) Again this number can be lower bounded (grossly) by $(1/(m+d+1))((m+ld+k+1)/(k+1))^{k+1}$ and we wish this to be at least as large as $n = h^k$, which follows easily. ■

Now it remains to mimic Lemma 4 in the multivariate setting.

CLAIM 9. *If $Q(\hat{x}, y)$ is a function satisfying (7) and $f(\hat{x})$ is a function satisfying (6) and $t > (m+ld)h^{k-1}$, then $(y - f(\hat{x}))$ divides $Q(\hat{x}, y)$.*

Proof (Sketch). Proof similar to that of Lemma 4. The function $p(\hat{x}) \stackrel{\text{def}}{=} Q(\hat{x}, f(\hat{x}))$ is a polynomial of degree $m+ld$ and identically zero. Hence $f(\hat{x})$ is a root of the polynomial $Q_{\hat{x}}(y) \stackrel{\text{def}}{=} Q(\hat{x}, y)$. ■

Thus with some optimization of parameters we are done.

THEOREM 10. *Given a table for a function $f' : H^k \rightarrow F$, where F is a field and H is an arbitrary finite subset of F , and integers t and d , a list of all polynomials $f : F^k \rightarrow F$ of degree at most d which agree with f' in t places can be found in time $\text{poly}(|H|^k)$ provided that*

$$\varepsilon \stackrel{\text{def}}{=} \frac{t}{|H|^k} > (k+1) \left(\frac{d+1}{|H|} \right)^{1/(k+1)}.$$

Proof. Let $h = |H|$. Set $m = 0$ and $l = (1/d)(k+1)(d+1)^{1/(k+1)}h^{k/(k+1)}$. Then since $m+ld \geq (k+1)(d+1)^{1/(k+1)}h^{k/(k+1)}$, Lemma 8 guarantees that Step B' of the multivariate algorithm will return a polynomial Q satisfying (7). Further, the condition on t implies that $t \geq (m+ld)h^{k-1}$, indicating that if a function f satisfies (6), then $y - f(\hat{x})$ will divide Q . Thus all polynomials will be returned by this procedure.

6. CONCLUSIONS

We first discuss the univariate reconstruction algorithm. The limit on t , i.e., $t \gtrsim \sqrt{2nd}$ is a significant weakness in the practical usability of this code. Many applications tend to work with $[n, k, \delta]$ codes where $\delta/n \rightarrow 0$, and for such growth our proof does not yield anything interesting. However, low-rate codes, and even Hadamard codes, have been used in the past² and when it suffices to use a low-rate code the reconstruction algorithm has some advantage. In Table I, we list the error which the reconstruction algorithm can tolerate and the number of solutions produced for such rates. The τ -reconstruction problem remains open

²Apparently the Mariner used a Hadamard code ([18], page 26).

for values of τ closer to the distance of the code, i.e., for $\tau = (1 - \varepsilon)\delta$ where $\varepsilon > 0$, and finding such a solution may be of both theoretical and practical interest. One does not expect the running time of such an algorithm to be polynomial in $1/\varepsilon$, since Goldreich *et al.* [11] give a NP-hardness proof in this case. The NP-hardness result there has $n = 2d + 3$, $t = d + 2$, with d being the degree of the desired polynomial. The instance is defined over the reals/rationals.

One of the main hopes in investigating this problem was that the algorithmic solution will be of some use in complexity theory. It is in this area that numerous applications for “Reed–Solomon-like” codes have occurred repeatedly. Examples include: (1) determining the hardness of the permanent on random instances [7, 16], (2) fault tolerant computing in distributed computing environments [3], and (3) many results involving “probabilistically-checkable proofs.” (See the survey by Feigenbaum [8] for a detailed look at many connections.) In particular, in the case of applications to probabilistically checkable proofs, it becomes useful to be able to characterize functions that are not close to polynomials (to be able to refute claimed proofs of incorrect statements). Note that our algorithm proves that the problem of recognizing points which are not very close to *any* polynomial is decidable in NP (by showing it is in P). The witness for this property is a proof that a function does not have a low-degree polynomial describing it on even a tiny fraction of the input. Recent work by Arora and Sudan [2] seems to justify our initial hope by using some of this analysis in a new analysis of a low-degree testing, which in turn leads to new constructions of proof systems. Another application has also been shown (very recently) by Impagliazzo and Nisan [12] in the area of cryptography. They use the algorithm presented here to obtain “random-efficient constructions of hardcore predicates for one-way functions.” (Defining any of these terms is beyond the scope of this paper.)

Now, moving on to the multivariate reconstruction problem, several glaring open problems remain. For starters, it is not clear how to set a bound on the number of solutions when $\varepsilon \stackrel{\text{def}}{=} t/|H|^k$ grows slower than $o(\sqrt{d/|H|})$. Even in the cases where ε is larger than $\Omega(\sqrt{d/|H|})$, the reconstruction problem is not fully solved. In particular, even for the case $k = 2$, no algorithm appears to solve this problem efficiently. This is hopefully an oversight on our part and some simple modification of the methods in [1, 9, 11] may work. However, such a solution does not remain algebraic. The question of whether there is an algebraic solution to the k -variate problem for general k , which works whenever $\varepsilon = \Omega(\sqrt{d/|H|})$, seems to be another interesting question.

Finally, we speculate on the complexity of the maximum-likelihood problem (or the nearest codeword problem). This problem is known to be NP-hard for general linear codes [5]. The hardness of the problem considered in [5] could be due to one of two reasons: (1) it is a maximum likelihood decoding problem rather than a τ -error correction problem; or (2) the code is specified as part of the input, rather than being a “well-known” one which is more

standard. It would be nice to know which of the two causes is responsible for the harness, since for all well-known codes the τ -error correction problem seems to be well solved. Bruck and Naor [6] present a code (not well known, but nevertheless easily presented) for which they show that the existence of small size maximum likelihood decoding circuits would imply the collapse of the polynomial hierarchy (using a result of Karp and Lipton's [14]). However, the codes presented by Bruck and Naor do not have a large distance. It still remains open if the maximum likelihood decoding problem is hard for any constant distance code. The Reed Solomon codes would have formed a good candidate to show the hardness of this problem, except that it is not as hard to solve. It would be nice to find another candidate for such a hardness result. Finally, there is still the possibility that some error-correcting codes are hard to decode, even to the full extent of their error-correction capacity. Alternately, we can ask the question: Is it possible to construct a τ -error correction algorithm for every τ -error correcting linear code, specified by its generator matrix? A positive answer to this might necessitate an algorithm to determine the distance of a code, a well-known open problem.

ACKNOWLEDGMENTS

I am grateful to Sanjeev Arora, Oded Goldreich, Ronitt Rubinfeld, Martin Tompa, and Shmuel Winograd for many valuable comments and discussions. I am especially grateful to Greg Sorkin for spending numerous hours implementing a version of the algorithm presented here.

REFERENCES

1. Ar, S., Lipton, R., Rubinfeld, R., and Sudan, M. (1992), Reconstructing algebraic functions from mixed data, in "Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, 1992," pp. 503–512.
2. Arora, S., and Sudan, M. (Aug. 1996), manuscript.
3. Ben-Or, M., Goldwasser, S., and Wigderson, A. (1988), Completeness theorems for non-cryptographic fault-tolerant distributed computation, in "Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988," pp. 1–10.
4. Berlekamp, E., and Welch, L. Error Correction of Algebraic Block Codes, U.S. Patent Number 4,633,470, 1986.
5. Berlekamp, E., McEliece, R., and Van Tilborg, H. On the inherent intractability of certain coding problems, *IEEE Trans. Inform. Theory* (1978), 384–386.
6. Bruck, J., and Naor, M. The hardness of decoding linear codes with preprocessing, *IEEE Trans. Inform. Theory* (1990), 381–385.
7. Feige, U., and Lund, C. On the hardness of computing the permanent of random matrices, in "Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992," pp. 643–654.

8. Feigenbaum, J. The use of coding theory in computational complexity, in "Proceedings of Symposia in Applied Mathematics," (R. Calderbank, Ed.), pp. 203–229, American Math Society, Providence, RI, 1995.
9. Gemmell, P., and Sudan, M. Highly resilient correctors for polynomials, *Inform. Process. Lett.* **43** (1992), 169–174.
10. Goldreich, O., and Levin, L. A. A hard-core predicate for any one-way function, in "Proceedings of the 21st Annual ACM Symposium on Theory of Computing, 1989," pp. 25–32.
11. Goldreich, O., Rubinfeld, R., and Sudan, M. Learning polynomials with queries: The highly noisy case (extended abstract), in "Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science, 1995," pp. 294–303.
12. Impagliazzo, R., and Nisan, N. personal communication, Sept. 1996.
13. Kaltofen, E. Polynomial factorization 1987–1991, in "LATIN '92," (I. Simon, Ed.), Lecture Notes in Computer Science, Vol. 585, pp. 294–313, Springer-Verlag, New York/Berlin, 1992.
14. Karp, R., and Lipton, R. Some connections between nonuniform and uniform complexity classes, in "Proceedings of the 12th Annual ACM Symposium on Theory of Computing, 1980," pp. 302–309.
15. Kushilevitz, E., and Mansour, Y. Learning decision trees using the Fourier spectrum, in "Proceedings of the 23th Annual ACM Symposium on Theory of Computing, 1991," pp. 455–464.
16. Lipton, R. New directions in testing, in "Distributed Computing and Cryptography," DIMACS Series in Discrete Math. and Theoretical Computer Science, Vol. 2, AMS, Providence, RI, 1991.
17. MacWilliams, F., and Sloane, N. "The Theory of Error-Correcting Codes," North-Holland, Amsterdam, 1981.
18. Van Lint, J. H. "Introduction to Coding Theory," Springer-Verlag, New York/Berlin, 1982.